Listing of Claims

1. (Canceled)

21. (Previously presented) A method for parsing in a distributed directory-enabled application environment using an eXtensible Markup Language ("XML") application program interface, the interface including a class factory, the method comprising:

accepting an XML file as an input stream,

parsing the input stream,

scanning the input stream for an object,

determining whether the object references a system service,

dynamically loading the service if referenced,

dynamically configuring the service, and

instantiating the object in the class factory, so that the service referenced by the object in the XML stream is automatically available to the object.

22. (Previously presented) The method of claim 21 further including determining whether the service is available before dynamically loading and configuring the service.

23. (Previously presented) The method of claim 22 further including determining whether the service is already loaded before loading the service.

24. (Previously presented) The method of claim 22 further including:

determining if the service is available; and

defaulting the object to a document object model during instantiation in the class factory if the service is unavailable.

25. (Previously presented) The method of claim 24 further including:

determining if there is a suitable document object model; and

defaulting the object to a highest available class during instantiation in the class factory if there is no suitable document object model.

26. (Previously presented) The method of claim 21 further including scanning the input stream for a plurality of objects.

27. (Previously presented) The method of claim 21 further including accepting a plurality of XML files as the input stream.

28. (Previously presented) A computer system for parsing in a distributed directory-enabled application environment using an eXtensible Markup Language ("XML") application program interface, the interface including a class factory, the system comprising:

at least one processor;

at least one memory accessible to the processor;

an application stored in a portion of the memory; and

software for parsing an XML file for the application, the software comprising instructions for:

accepting the XML file as an input stream,

parsing the input stream,

scanning the input stream for an object,

determining whether the object references a system service,

dynamically loading the service if referenced,

dynamically configuring the service, and

instantiating the object in the class factory, so that the service referenced by the object in the XML stream is automatically available to the object..

29. (Previously presented) The system of claim 28, wherein the software further includes instructions for:

determining whether the service is available before dynamically configuring and loading the service.

30. (Previously presented) The system of claim 29, wherein the software further includes instructions for:

determining whether the service is already loaded before loading the service.

31. (Previously presented) The system of claim 29, wherein the software further includes instructions for:

determining if the service is available; and

defaulting the object to a document object model during instantiation in the class factory if the service is unavailable.

32. (Previously presented) The system of claim 31, wherein the software further includes instructions for:

determining if there is no suitable document object model; and

defaulting the object to a highest available class during instantiation in the class factory if there is no suitable document object model.

33. (Previously presented) The system of claim 28, wherein the software further includes instructions for:

scanning the input stream for a plurality of objects.

34. (Previously presented) The system of claim 28, wherein the software further includes instructions for:

accepting a plurality of XML files as the input stream.

35. (Previously presented) A method for parsing in a distributed directory-enabled application environment using an eXtensible Markup Language ("XML") application program interface, the interface including a class factory, the method comprising:

parsing an XML input file to identify an object;

determining whether the object references a system service;

if the service is referenced and not loaded, dynamically loading the service;

dynamically configuring the service; and

instantiating the object in the class factory, so that the service referenced by the object in the XML stream is automatically available to the object.